

A Match in Time Saves Nine: Deterministic Online Matching With Delays*

Marcin Bienkowski¹, Artur Kraska¹, and Paweł Schmidt¹

¹ Institute of Computer Science, University of Wrocław, Poland

Abstract

We consider the problem of online Min-cost Perfect Matching with Delays (MPMD) introduced by Emek et al. (STOC 2016). In this problem, an even number of requests appear in a metric space at different times and the goal of an online algorithm is to match them in pairs. In contrast to traditional online matching problems, in MPMD all requests appear online and an algorithm can match any pair of requests, but such decision may be delayed (e.g., to find a better match). The cost is the sum of matching distances and the introduced delays.

We present the first deterministic online algorithm for this problem. Its competitive ratio is $O(m^{\log_2 5.5}) = O(m^{2.46})$, where $2m$ is the number of requests. This is polynomial in the number of metric space points if all requests are given at different points. In particular, the bound does not depend on other parameters of the metric, such as its aspect ratio. Unlike previous (randomized) solutions for the MPMD problem, our algorithm does not need to know the metric space in advance.

1998 ACM Subject Classification F.1.2 Modes of Computation: Online computation, F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases online matching, delays, rent-or-buy, competitive analysis

1 Introduction

In this paper, we give a deterministic online algorithm for the problem of Min-cost Perfect Matching with Delays (MPMD) [22, 5]. For an informal description, imagine that there are human players who are logging in real time into a gaming website, each wanting to play chess against another human player. The system pairs the players according to their known capabilities, such as playing strength. A decision with whom to match a given player can be delayed until a reasonable match is found. That is, the website tries to simultaneously minimize two objectives: the waiting times of players and their dissimilarity, i.e., each player would like to play with another one with similar capabilities. An algorithm running the website has to work online, without the knowledge about future player arrivals and make its decision irrevocably: once two players are paired, they remain paired forever.

1.1 Problem definition

More formally, in the MPMD problem there is a metric space \mathcal{X} with a distance function $\text{dist} : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, both known from the beginning to an online algorithm. An online part of the input is a sequence of $2m$ requests $\{(p_i, t_i)\}_{i=1}^{2m}$, where point $p_i \in \mathcal{X}$ corresponds to a player in our informal description above and t_i is the time of its arrival. Clearly, $t_1 \leq t_2 \leq \dots \leq t_{2m}$. The integer m is not known a priori to an online algorithm. At any

* Partially supported by Polish National Science Centre grant 2016/22/E/ST6/00499.



© Marcin Bienkowski, Artur Kraska, Paweł Schmidt;
licensed under Creative Commons License CC-BY

Leibniz International Proceedings in Informatics
LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

time τ , an online algorithm may decide to match any pair of requests (p_i, t_i) and (p_j, t_j) that have already arrived ($\tau \geq t_i$ and $\tau \geq t_j$) and have not been matched yet. The cost incurred by such *matching edge* is $\text{dist}(p_i, p_j) + (\tau - t_i) + (\tau - t_j)$, i.e., is the sum of the *connection cost* and the *waiting costs* of these two requests.

The goal is to eventually match all requests and minimize the total cost. We use a typical yardstick to measure the performance: a competitive ratio [13], defined as the maximum, over all inputs, of the ratios between the cost of an online algorithm and the cost of an optimal offline solution OPT that knows the entire input sequence in advance.

1.2 Previous work

The MPMD problem was introduced by Emek et al. [22], who presented a randomized $O(\log^2 n + \log \Delta)$ -competitive algorithm. There, n is the number of points in the metric space \mathcal{X} and Δ is its aspect ratio (the ratio between the largest and the smallest distance in \mathcal{X}). The competitive ratio was subsequently improved by Azar et al. [5] to $O(\log n)$. They showed that the ratio of any randomized algorithm is at least $\Omega(\sqrt{\log n})$. The currently best lower bound of $\Omega(\log n / \log \log n)$ for randomized solutions was given by Ashlagi et al. [3].

So far, the construction of a competitive *deterministic* algorithm for general metric spaces remained an open problem. It was hypothesized that competitive ratios achievable by deterministic algorithms might be superpolynomial in n (cf. Section 5 of [5]). Deterministic algorithms were known only for simple spaces: Azar et al. [5] gave an $O(\text{height})$ -competitive algorithm for trees and Emek et al. [23] constructed a 3-competitive deterministic solution for two-point metric (the competitive ratio is best possible for such metric).

1.3 Our contribution

In this paper, we give the first deterministic algorithm for any metric space, whose competitive ratio is $O(m^{\log_2 5.5}) = O(m^{2.46})$, where $2m$ is the number of requests. Typically, for our gaming application, m is smaller than n (although in full generality it can be also larger if multiple requests arrive at the same point of the metric space \mathcal{X}). While previous solutions to the MPMD problem [22, 5] required \mathcal{X} to be finite and known a priori (to approximate it first by a random HST tree [24] or a random HST tree with reduced height [8]), our solution works even when \mathcal{X} is revealed in online manner. That is, we require only that, together with any request r , an online algorithm learns the distances from r to all previous, not yet matched requests.

Our online algorithm ALG uses a simple, local, semi-greedy scheme to find a suitable matching pair. In the analysis, we fix a final perfect matching of OPT and observe what happens when we gradually add matching edges that ALG creates during its execution. That is, we trace the evolution of alternating paths and cycles in time. To bound the cost of ALG, we charge the cost of an edge that ALG is adding against the cost of already existing matching edges from the same alternating path. Interestingly, our charging argument on alternating cycles bears some resemblance to the analyses of algorithms for the problems that are not directly related to MPMD: online metric (bipartite) matching on line metrics [2] and offline greedy matching [40].

1.4 Related work

Originally, matching problems have been studied in variants where delaying decisions was not permitted. The setting most similar to the MPMD problem is called online *metric bipartite*

matching. It involves m *offline points* given to an algorithm at the beginning and m *requests* presented in online manner that need to be matched (immediately after their arrival) to offline points. Both points and requests lie in a common metric space and the goal is to minimize the weight of a perfect matching created by an algorithm. For general metric spaces, the best randomized solution is $O(\log m)$ -competitive [7, 26, 37], and the deterministic algorithms achieve the optimal competitive ratio of $2m - 1$ [27, 32]. Interestingly, even for line metrics [2, 25, 33], the best known deterministic algorithm attains a competitive ratio that is polynomial in m [2].

In comparison, in the MPMD problem considered in this paper, all $2m$ requests appear in online manner, m is not known to an algorithm, and we allow to match any pair of them. That said, there is also a bipartite variant of the MPMD problem, in which all requests appear online, but m of them are negative and m are positive. An algorithm may then only match pairs of requests of different polarities [4, 3].

The MPMD problem can be cast as augmenting min-cost perfect matching with a time axis, allowing the algorithm to delay its decisions, but penalizing the delays. There are many other problems that use this paradigm: most notably the ski-rental problem and its continuous counterpart, the spin-block problem [29], where a purchase decision can be delayed until renting cost becomes sufficiently large. Such rent-or-buy (wait-or-act) trade-offs are also found in other areas, for example in aggregating messages in computer networks [1, 11, 21, 28, 31, 39], in aggregating orders in supply-chain management [9, 10, 14, 15, 17, 18] or in some scheduling variants [6].

Finally, there is a vast amount of work devoted to other online matching variant, where offline points and online requests are connected by graph edges and the goal is to maximize the weight or the cardinality of the produced matching. These types of matching problems have been studied since the seminal work of Karp et al. [30] and are motivated by applications to online auctions [12, 16, 19, 20, 30, 34, 36, 38]. They were also studied under stochastic assumptions on the input, see, e.g., a survey by Mehta [35].

2 Algorithm

We will identify requests with the points at which they arrive. To this end, we assume that all requested points are different, but we allow distances between different metric points to be zero. For any request p , we denote the time of its arrival by $\text{atime}(p)$.

Our algorithm is parameterized with real numbers $\alpha > 0$ and $\beta > 1$, whose exact values will be optimized later. For any request p , we define its waiting time at time $\tau \geq \text{atime}(p)$ as

$$\text{wait}_\tau(p) = \tau - \text{atime}(p)$$

and its budget at time τ as

$$\text{budget}_\tau(p) = \alpha \cdot \text{wait}_\tau(p) .$$

Our online algorithm ALG matches two requests p and q at time τ as soon as the following two conditions are satisfied.

- *Budget sufficiency*: $\text{budget}_\tau(p) + \text{budget}_\tau(q) \geq \text{dist}(p, q)$.
- *Budget balance*: $\text{budget}_\tau(p) \leq \beta \cdot \text{budget}_\tau(q)$ and $\text{budget}_\tau(q) \leq \beta \cdot \text{budget}_\tau(p)$.

Note that the budget balance condition is equivalent to relations on waiting times, i.e., $\text{wait}_\tau(p) \leq \beta \cdot \text{wait}_\tau(q)$ and $\text{wait}_\tau(q) \leq \beta \cdot \text{wait}_\tau(p)$.

If the conditions above are met simultaneously for many point pairs, we break ties arbitrarily, and process them in any order. Note that at the time when p and q become

matched, the sum of their budgets may exceed $\text{dist}(p, q)$. For example, this occurs when q appears at time strictly larger than $\text{atime}(p) + \text{dist}(p, q)$: they are then matched by ALG as soon as the budget balance condition becomes true.

The observation below follows immediately by the definition of ALG.

► **Observation 1.** Fix time τ and two requests p and q , such that $\text{atime}(p) \leq \tau$ and $\text{atime}(q) \leq \tau$. Assume that neither p nor q has been matched by ALG strictly before time τ . Then exactly one of the following conditions holds:

- $\alpha \cdot (\text{wait}_\tau(p) + \text{wait}_\tau(q)) \leq \text{dist}(p, q)$,
- $\alpha \cdot (\text{wait}_\tau(p) + \text{wait}_\tau(q)) > \text{dist}(p, q)$ and $\text{wait}_\tau(p) \geq \beta \cdot \text{wait}_\tau(q)$,
- $\alpha \cdot (\text{wait}_\tau(p) + \text{wait}_\tau(q)) > \text{dist}(p, q)$ and $\text{wait}_\tau(q) \geq \beta \cdot \text{wait}_\tau(p)$.

3 Analysis

To analyze the performance of ALG, we look at matchings generated by ALG and by an optimal offline algorithm OPT. If points p and q were matched at time τ by ALG, then we say that ALG creates a (matching) edge $e = (p, q)$. Its cost is

$$\text{cost}_{\text{ALG}}(e) = \text{cost}_{\text{ALG}}(p, q) = \text{dist}(p, q) + \text{wait}_\tau(p) + \text{wait}_\tau(q) .$$

We call e an ALG-edge. The cost_{OPT} of an edge in the solution of OPT (an OPT-edge) is defined analogously. In an optimal solution, however, the matching time is always equal to the arrival time of the later of two matched requests.

We now consider a dynamically changing graph consisting of requested points, OPT-edges and ALG-edges. For the analysis, we assume that it changes in the following way: all requested points and all OPT-edges are present in the graph from the beginning, but the ALG-edges are added to the graph in m steps, in the order they are created by ALG.

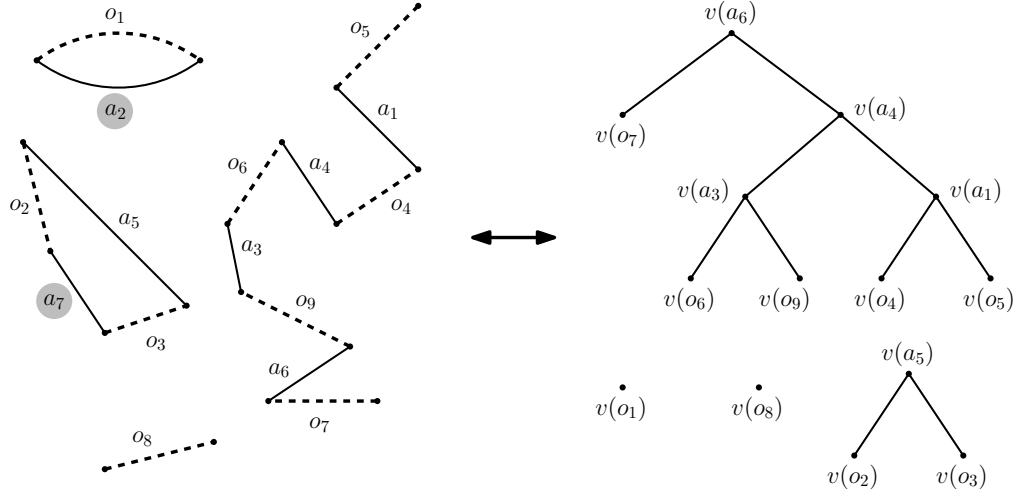
At all times, the matching edges present in the graph form alternating paths or cycles (i.e., paths or cycles whose edges are interleaved ALG-edges and OPT-edges). Furthermore, any node-maximal alternating path starts and ends with OPT-edges. Assume now that a matching edge e created by ALG is added to the graph. It may either connect the ends of two different alternating paths, thus creating a single longer alternating path or connect the ends of one alternating path, generating an alternating cycle. In the former case, we call edge e *non-final*, in the latter case — *final*. Note that at the end of the ALG execution, when m ALG-edges are added, the graph contains only alternating cycles.

We extend the notion of cost to alternating path and cycles. For any cycle C , $\text{cost}(C)$ is simply the sum of costs of its edges: the cost of an OPT-edge on such cycle is the cost paid by OPT and the cost of an ALG-edge is that of ALG. We also define $\text{cost}_{\text{OPT}}(C)$, $\text{cost}_{\text{ALG}}(C)$ and $\text{cost}_{\text{ALG-NF}}(C)$ as the costs of OPT-edges, ALG-edges and non-final ALG-edges on cycle C , respectively. Clearly, $\text{cost}_{\text{ALG}}(C) + \text{cost}_{\text{OPT}}(C) = \text{cost}(C)$. We define the same notions for alternating paths; as a path P does not contain final ALG-edges, $\text{cost}_{\text{ALG-NF}}(P) = \text{cost}_{\text{ALG}}(P)$.

An alternating path is called κ -step maximal alternating path if it exists in the graph after ALG matched κ pairs and it cannot be extended, i.e., it ends with two requests that are not yet matched by the first κ ALG-edges.

3.1 Tree construction

To facilitate the analysis, along with the graph, we create a dynamically changing forest F of binary trees, where each leaf of F corresponds to an OPT-edge and each internal (non-leaf)



■ **Figure 1** The left side contains an example graph consisting of all OPT-edges o_1, o_2, \dots, o_9 (dashed lines) and the first $\kappa = 7$ ALG-edges a_1, a_2, \dots, a_7 (solid lines). ALG-edges are numbered in the order they were created and added to the graph. Shaded ALG-edges (a_2 and a_7) are final, the remaining ones are non-final.

The right side depicts the corresponding forest F : leaves of F represent OPT-edges and non-leaf nodes of F correspond to non-final ALG-edges. Trees rooted at nodes $v(o_1)$ and $v(a_5)$ represent alternating cycles and those rooted at nodes $v(a_6)$ and $v(o_8)$ represent alternating paths in the graph.

node of F to a non-final ALG-edge (and vice versa). After ALG matched κ pairs, each subtree of F corresponds to a κ -step maximal alternating path or to an alternating cycle. More precisely, at the beginning, F consists of m single nodes representing OPT-edges. Afterwards, whenever an ALG-edge is created, we perform the following operation on F .

- When a non-final ALG-edge $e = (p, q)$ is added to the graph, we look at the two alternating paths P and Q that end with p and q , respectively. We take the corresponding trees $T(P)$ and $T(Q)$ of F . We add a node $v(e)$ (representing edge e) to F and make $T(P)$ and $T(Q)$ its subtrees.
- When a final ALG-edge $e = (p, q)$ is added to the graph, it turns an alternating path P into an alternating cycle C . We then simply say that the tree $T(P)$ that corresponded to P , now corresponds to C .

An example of the graph and the associated forest F is presented in [Figure 1](#).

For any tree node w , we define its weight $\text{weight}(w)$ as the cost of the corresponding matching edge, i.e., the cost of an OPT-edge for a leaf and the cost of a non-final ALG-edge for a non-leaf node. For any node w , by T_w we denote the tree rooted at w . We extend the notion of weight in a natural manner to all subtrees of F . In these terms, the weight of a tree T in F is equal to the total cost of the corresponding alternating path. (If T represents an alternating cycle C , then its weight is equal to the cost of C minus the cost of the final ALG-edge from C .)

Note that we consistently used terms “points” and “edges” for objects that ALG and OPT are operating on in the metric space \mathcal{X} . On the other hand, the term “nodes” will always refer to tree nodes in F and we will not use the term “edge” to denote an edge in F .

3.2 Outline of the analysis

Our approach to bounding the cost of ALG is now as follows. We look at the forest F at the end of ALG execution. The corresponding graph contains only alternating cycles. The cost of non-final ALG-edges is then, by the definition, equal to the total weight of internal (non-leaf) nodes of F , while the cost of OPT-edges is equal to the total weight of leaves of F . Hence, our goal is to relate the total weight of any tree to the weight of its leaves.

The central piece of our analysis is showing that for any internal node w with children u and v , it holds that $\text{weight}(w) \leq \xi \cdot \min\{\text{weight}(T_u), \text{weight}(T_v)\}$, where ξ is a constant depending on parameters α and β (see [Corollary 4](#)). Using this relation, we will bound the total weight of any tree by $O(m^{\log_2(\xi+2)-1})$ times the total weight of its leaves. This implies the same bound on the ratio between non-final ALG-edges and OPT-edges on each alternating cycle.

Finally, we show that the cost of final ALG-edges incurs at most an additional constant factor in the total cost of ALG.

3.3 Cost of non-final ALG-edges

As described in [Section 3.1](#), when ALG adds a κ -th ALG-edge e to the graph, and this edge is non-final, e joins two $(\kappa - 1)$ -step maximal alternating paths P and Q . We will bound $\text{cost}_{\text{ALG}}(e)$ by a constant (depending on α and β) times $\min\{\text{cost}(P), \text{cost}(Q)\}$. We start with bounding the waiting cost of ALG related to one endpoint of e .

► **Lemma 2.** *Let $e = (p, q)$ be the κ -th ALG-edge added at time τ , such that e is non-final. Let $P = (a_1, a_2, \dots, a_\ell)$ be the $(\kappa - 1)$ -step maximal alternating path ending at $p = a_1$. Then, $\text{wait}_\tau(p) \leq \max\{\alpha^{-1}, \beta/(\beta - 1)\} \cdot \text{cost}(P)$.*

Proof. First we lower-bound the cost of an alternating path P . We look at any edge (a_i, a_{i+1}) from P . Its cost (no matter whether paid by ALG or OPT) is certainly larger than $\text{dist}(a_i, a_{i+1}) + |\text{atime}(a_i) - \text{atime}(a_{i+1})|$. Therefore, using triangle inequality (on distances and times), we obtain

$$\begin{aligned} \text{cost}(P) &\geq \sum_{i=1}^{\ell-1} (\text{dist}(a_i, a_{i+1}) + |\text{atime}(a_i) - \text{atime}(a_{i+1})|) \\ &\geq \text{dist}(a_1, a_\ell) + |\text{atime}(a_1) - \text{atime}(a_\ell)|. \end{aligned} \quad (1)$$

Therefore, in our proof we will simply bound $\text{wait}_\tau(p) = \text{wait}_\tau(a_1)$ using either $\text{dist}(a_1, a_\ell)$ or $|\text{atime}(a_1) - \text{atime}(a_\ell)|$.

Recall that ALG matches a_1 at time τ . Consider the state of a_ℓ at time τ . If a_ℓ has not been presented to ALG yet ($\text{atime}(a_\ell) > \tau$), then $\text{wait}_\tau(a_1) = \tau - \text{atime}(a_1) < \text{atime}(a_\ell) - \text{atime}(a_1) < \beta/(\beta - 1) \cdot (\text{atime}(a_\ell) - \text{atime}(a_1))$, and the lemma follows.

In the remaining part of the proof, we assume that a_ℓ was already presented to the algorithm ($\text{atime}(a_\ell) \leq \tau$). As P is a $(\kappa - 1)$ -step maximal alternating path, a_ℓ is not matched by ALG right after ALG creates $(\kappa - 1)$ -th matching edge. The earliest time when a_ℓ may become matched is when ALG creates the next, κ -th matching edge, i.e., at time τ . Therefore a_ℓ is not matched before time τ .

Now observe that there must be a reason for which requests a_1 and a_ℓ have not been matched with each other before time τ . Roughly speaking, either the sum of budgets of requests a_1 and a_ℓ does not suffice to cover the cost of $\text{dist}(a_1, a_\ell)$ or one of them waits significantly longer than the other. Formally, we apply [Observation 1](#) to pair (a_1, a_ℓ) obtaining three possible cases. In each of the cases we bound $\text{wait}_\tau(a_1)$ appropriately.

Case 1 (insufficient budgets). If $\alpha \cdot (\text{wait}_\tau(a_1) + \text{wait}_\tau(a_\ell)) \leq \text{dist}(a_1, a_\ell)$, then by non-negativity of $\text{wait}_\tau(a_\ell)$, it follows that $\text{wait}_\tau(a_1) \leq \alpha^{-1} \cdot \text{dist}(a_1, a_\ell)$.

Case 2 (a_1 waited much longer than a_ℓ). If $\alpha \cdot (\text{wait}_\tau(a_1) + \text{wait}_\tau(a_\ell)) > \text{dist}(a_1, a_\ell)$ and $\text{wait}_\tau(a_1) \geq \beta \cdot \text{wait}_\tau(a_\ell)$, then $\text{atime}(a_\ell) - \text{atime}(a_1) = \text{wait}_\tau(a_1) - \text{wait}_\tau(a_\ell) \geq (1 - 1/\beta) \cdot \text{wait}_\tau(a_1)$. Therefore, $\text{wait}_\tau(a_1) \leq \beta/(\beta - 1) \cdot |\text{atime}(a_1) - \text{atime}(a_\ell)|$.

Case 3 (a_ℓ waited much longer than a_1). If $\alpha \cdot (\text{wait}_\tau(a_1) + \text{wait}_\tau(a_\ell)) > \text{dist}(a_1, a_\ell)$ and $\text{wait}_\tau(a_\ell) \geq \beta \cdot \text{wait}_\tau(a_1)$, then $\text{atime}(a_1) - \text{atime}(a_\ell) = \text{wait}_\tau(a_\ell) - \text{wait}_\tau(a_1) \geq (\beta - 1) \cdot \text{wait}_\tau(a_1)$. Thus, $\text{wait}_\tau(a_1) \leq 1/(\beta - 1) \cdot |\text{atime}(a_1) - \text{atime}(a_\ell)| < \beta/(\beta - 1) \cdot |\text{atime}(a_1) - \text{atime}(a_\ell)|$. \blacktriangleleft

► **Lemma 3.** Let $e = (p, q)$ be the κ -th ALG-edge, such that e is non-final. Let $P = (a_1, a_2, \dots, a_\ell)$ and $Q = (b_1, b_2, \dots, b_{\ell'})$ be the $(\kappa - 1)$ -step maximal alternating path ending at $p = a_1$ and $q = b_1$, respectively. Then,

$$\text{cost}_{\text{ALG}}(e) \leq (1 + \alpha) \cdot (\beta + 1) \cdot \max\{\alpha^{-1}, \beta/(\beta - 1)\} \cdot \min\{\text{cost}(P), \text{cost}(Q)\} .$$

Proof. Let τ be the time when p is matched with q by ALG. Using the definition of cost_{ALG} , we obtain

$$\begin{aligned} \text{cost}_{\text{ALG}}(p, q) &= \text{dist}(p, q) + \text{wait}_\tau(p) + \text{wait}_\tau(q) \\ &\leq \text{budget}_\tau(p) + \text{budget}_\tau(q) + \text{wait}_\tau(p) + \text{wait}_\tau(q) \\ &= (1 + \alpha) \cdot (\text{wait}_\tau(p) + \text{wait}_\tau(q)) \\ &\leq (1 + \alpha) \cdot (\beta + 1) \cdot \min\{\text{wait}_\tau(p), \text{wait}_\tau(q)\} . \end{aligned} \quad (2)$$

The first inequality follows by the budget sufficiency condition of ALG and the second one by the budget balance condition.

By Lemma 2, $\text{wait}_\tau(p) \leq \max\{\alpha^{-1}, \beta/(\beta - 1)\} \cdot \text{cost}(P)$ and $\text{wait}_\tau(q) \leq \max\{\alpha^{-1}, \beta/(\beta - 1)\} \cdot \text{cost}(Q)$, which combined with (2) immediately yield the lemma. \blacktriangleleft

Recall now the iterative construction of forest F from Section 3.1: whenever a non-final matching edge e created by ALG joins two alternating paths P and Q , we add a new node w to F , such that $\text{weight}(w) = \text{cost}_{\text{ALG}}(e)$ and make trees $T(P)$ and $T(Q)$ its children. These trees correspond to paths P and Q , and satisfy $\text{weight}(T(P)) = \text{cost}(P)$ and $\text{weight}(T(Q)) = \text{cost}(Q)$. Therefore, Lemma 3 immediately implies the following equivalent relation on tree weights.

► **Corollary 4.** Let w be an internal node of the forest F whose children are u and v . Then, $\text{weight}(w) \leq (1 + \alpha) \cdot (\beta + 1) \cdot \max\{\alpha^{-1}, \beta/(\beta - 1)\} \cdot \min\{\text{weight}(T_u), \text{weight}(T_v)\}$.

This relation can be used to express the total weight of a tree of F in terms of the total weight of its leaves. The proof of the following technical lemma is deferred to Section 4. Here, we present how to use it to bound the cost of ALG on non-final edges of a single alternating cycle.

► **Lemma 5.** Let T be a weighted full binary tree and $\xi \geq 0$ be any constant. Assume that for each internal node w with children u and v , their weights satisfy $\text{weight}(w) \leq \xi \cdot \min\{\text{weight}(T_u), \text{weight}(T_v)\}$. Then,

$$\text{weight}(T) \leq (\xi + 2) \cdot |L(T)|^{\log_2(\xi/2+1)} \cdot \text{weight}(L(T)) ,$$

where $L(T)$ is the set of leaves of T and $\text{weight}(L(T))$ is their total weight.

► **Lemma 6.** *Let C be an alternating cycle obtained from combining matchings of ALG and OPT . Then $\text{cost}_{\text{ALG-NF}}(C) \leq (\xi + 2) \cdot m^{\log_2(\xi/2+1)} \cdot \text{cost}_{\text{OPT}}(C)$, where $\xi = (1 + \alpha) \cdot (\beta + 1) \cdot \max\{\alpha^{-1}, \beta/(\beta - 1)\}$.*

Proof. As described in Section 3.1, C is associated with a tree T from forest F , such that OPT -edges of C correspond to the set of leaves of T (denoted $L(T)$) and non-final ALG -edges of C correspond to internal (non-leaf) nodes of T . Hence, $\text{cost}_{\text{OPT}}(C) = \text{weight}(L(T))$ and $\text{cost}_{\text{ALG-NF}}(C) + \text{cost}_{\text{OPT}}(C) = \text{weight}(T)$.

By Corollary 4, the weight of any internal tree node w with children u, v satisfies $\text{weight}(w) \leq \xi \cdot \min\{\text{weight}(T_u), \text{weight}(T_v)\}$. Therefore, we may apply Lemma 5 to tree T , obtaining $\text{weight}(T) \leq (\xi + 2) \cdot |L(T)|^{\log_2(\xi/2+1)} \cdot \text{weight}(L(T))$, and thus

$$\begin{aligned} \text{cost}_{\text{ALG-NF}}(C) &\leq \text{weight}(T) \leq (\xi + 2) \cdot |L(T)|^{\log_2(\xi/2+1)} \cdot \text{weight}(L(T)) \\ &\leq (\xi + 2) \cdot m^{\log_2(\xi/2+1)} \cdot \text{weight}(L(T)) \\ &= (\xi + 2) \cdot m^{\log_2(\xi/2+1)} \cdot \text{cost}_{\text{OPT}}(C) . \end{aligned}$$

The last inequality follows as $|L(T)|$, the number of T leaves, is equal to the number of OPT -edges on cycle C , which is clearly at most m . ◀

3.4 Cost of final ALG-edges

In the previous section, we derived a bound on the cost of all non-final ALG -edges. The following lemma shows that the cost of final ALG -edges contribute at most a constant factor to the competitive ratio.

► **Lemma 7.** *Let e be a final ALG -edge matched at time τ and C be the alternating cycle containing e . Then $\text{cost}_{\text{ALG}}(e) \leq (1 + \alpha) \cdot \max\{\alpha^{-1}, (\beta + 1)/(\beta - 1)\} \cdot (\text{cost}_{\text{ALG-NF}}(C) + \text{cost}_{\text{OPT}}(C))$.*

Proof. Fix a final ALG -edge $e = (p, q)$, where $\text{atime}(q) \geq \text{atime}(p)$. By the budget sufficiency condition of ALG ,

$$\text{cost}_{\text{ALG}}(e) \leq (1 + \alpha) \cdot (\text{wait}_\tau(p) + \text{wait}_\tau(q)) . \quad (3)$$

Our goal now is to bound $\text{wait}_\tau(p) + \text{wait}_\tau(q)$ in terms of $\text{dist}(p, q)$ or $\text{atime}(q) - \text{atime}(p)$. Observe that whenever ALG matches two requests, the budget sufficiency condition of ALG or one of the inequalities of the budget balance condition is satisfied with equality. We apply this observation to pair (p, q) .

- If the budget sufficiency condition holds with equality, $\alpha \cdot (\text{wait}_\tau(p) + \text{wait}_\tau(q)) = \text{dist}(p, q)$, and therefore $\text{wait}_\tau(p) + \text{wait}_\tau(q) = \alpha^{-1} \cdot \text{dist}(p, q)$.
- If the budget balance condition holds with equality, $\beta \cdot \text{wait}_\tau(q) = \text{wait}_\tau(p)$. Then,

$$\begin{aligned} (\beta - 1) \cdot (\text{wait}_\tau(p) + \text{wait}_\tau(q)) &= (\beta - 1) \cdot (\beta + 1) \cdot \text{wait}_\tau(q) \\ &= (\beta + 1) \cdot (\text{wait}_\tau(p) - \text{wait}_\tau(q)) \\ &= (\beta + 1) \cdot (\text{atime}(q) - \text{atime}(p)) . \end{aligned}$$

Hence, in either case it holds that

$$\text{wait}_\tau(p) + \text{wait}_\tau(q) \leq \max\left\{\alpha^{-1}, \frac{\beta + 1}{\beta - 1}\right\} \cdot (\text{dist}(p, q) + |\text{atime}(q) - \text{atime}(p)|) . \quad (4)$$

Finally, we bound $\text{dist}(p, q) + |\text{atime}(q) - \text{atime}(p)|$ in terms of costs of other edges of C . These edges form a path $P = (a_1, a_2, \dots, a_\ell)$, where $a_1 = p$ and $a_\ell = q$. By the triangle inequality applied to distances and time differences (in the same way as in (1)), we obtain that

$$\text{dist}(p, q) + |\text{atime}(q) - \text{atime}(p)| \leq \text{cost}(P) = \text{cost}_{\text{ALG-NF}}(C) + \text{cost}_{\text{OPT}}(C) . \quad (5)$$

The lemma follows immediately by combining (3), (4) and (5). \blacktriangleleft

3.5 The competitive ratio

Finally, we optimize constants α and β used throughout the previous sections and bound the competitiveness of ALG.

► **Theorem 8.** *For $\beta = 2$ and $\alpha = 1/2$, the competitive ratio of ALG is $O(m^{\log_2 5.5}) = O(m^{2.46})$, where $2m$ is the number of requests in the input sequence.*

Proof. The union of matchings constructed by ALG and OPT can be split into a set \mathcal{C} of disjoint cycles. It is sufficient to show that we have the desired performance guarantee on each cycle from \mathcal{C} .

Fix a cycle $C \in \mathcal{C}$. Let $e = (p, q)$ be the final ALG-edge of C . By Lemma 7, $\text{cost}_{\text{ALG}}(e) \leq 4.5 \cdot (\text{cost}_{\text{ALG-NF}}(C) + \text{cost}_{\text{OPT}}(C))$. Therefore, the competitive ratio of ALG is at most

$$\frac{\text{cost}_{\text{ALG}}(C)}{\text{cost}_{\text{OPT}}(C)} \leq \frac{5.5 \cdot \text{cost}_{\text{ALG-NF}}(C) + 4.5 \cdot \text{cost}_{\text{OPT}}(C)}{\text{cost}_{\text{OPT}}(C)} \leq O(m^{\log_2 5.5}) = O(m^{2.46}) ,$$

where the second inequality follows by Lemma 6. \blacktriangleleft

4 Relating weights in trees (proof of Lemma 5)

We start with the following technical claim that will facilitate the inductive proof of Lemma 5.

► **Lemma 9.** *Fix any constant $\xi \geq 0$ and let $f(a) = a^{\log_2(\xi+2)}$. Then, $\xi \cdot \min\{f(x), f(y)\} + f(x) + f(y) \leq f(x+y)$ for all $x, y \geq 0$.*

Proof. Fix any $z \geq 0$ and let $g_z(a) = (\xi+1) \cdot f(a) + f(z-a)$. We observe that $g_z(0) = f(z)$ and $g_z(z/2) = (\xi+1) \cdot f(z/2) + f(z/2) = (\xi+2) \cdot (z/2)^{\log_2(\xi+2)} = z^{\log_2(\xi+2)} = f(z)$. Moreover, the function g_z is convex as it is a sum of two convex functions. As $g_z(0) = g_z(z/2) = f(z)$, by convexity, $g_z(a) \leq f(z)$ for any $a \in [0, z/2]$.

To prove the lemma, assume without loss of generality that $x \leq y$. By the monotonicity, $f(x) \leq f(y)$, and therefore

$$\begin{aligned} \xi \cdot \min\{f(x), f(y)\} + f(x) + f(y) &= (\xi+1) \cdot f(x) + f((x+y) - x) \\ &= g_{x+y}(x) \\ &\leq f(x+y) . \end{aligned}$$

The last inequality follows as $x \leq (x+y)/2$. \blacktriangleleft

Proof of Lemma 5. We scale weights of all nodes, so that the average weight of each leaf is 1, i.e., we define a scaled weight function ws as

$$\text{ws}(w) = \text{weight}(w) \cdot \frac{|\text{L}(T)|}{\text{weight}(\text{L}(T))} .$$

Note that ws also satisfies $\text{ws}(w) \leq \xi \cdot \min\{\text{ws}(T_u), \text{ws}(T_v)\}$. Moreover, since we scaled all weights in the very same way, $\text{ws}(T)/\text{ws}(\mathcal{L}(T)) = \text{weight}(T)/\text{weight}(\mathcal{L}(T))$, and hence to show the lemma, it suffices to bound the term $\text{ws}(T)/\text{ws}(\mathcal{L}(T))$.

For any node $w \in T$ and the corresponding subtree T_w rooted at w , we define $\text{size}(T_w) = \text{ws}(\mathcal{L}(T_w)) + |\mathcal{L}(T_w)|$. We inductively show that for any node of $w \in T$, it holds that

$$\text{ws}(T_w) \leq \text{size}(T_w)^{\log_2(\xi+2)} . \quad (6)$$

For the induction basis, assume that w is a leaf of T . Then,

$$\text{ws}(T_w) = \text{ws}(\mathcal{L}(T_w)) \leq \text{size}(T_w) \leq \text{size}(T_w)^{\log_2(\xi+2)} ,$$

where the last inequality follows as $\text{size}(T_w) \geq |\mathcal{L}(T_w)| = 1$ and $\xi > 0$.

For the inductive step, let w be a non-leaf node of T and let u and v be its children. Then,

$$\begin{aligned} \text{ws}(T_w) &= \text{ws}(T_u) + \text{ws}(T_v) + \text{ws}(w) \\ &\leq \text{ws}(T_u) + \text{ws}(T_v) + \xi \cdot \min\{\text{ws}(T_u), \text{ws}(T_v)\} \\ &\leq \text{size}(T_u)^{\log_2(\xi+2)} + \text{size}(T_v)^{\log_2(\xi+2)} + \xi \cdot \min\{\text{size}(T_u)^{\log_2(\xi+2)}, \text{size}(T_v)^{\log_2(\xi+2)}\} \\ &\leq (\text{size}(T_u) + \text{size}(T_v))^{\log_2(\xi+2)} \\ &= \text{size}(T_w)^{\log_2(\xi+2)} . \end{aligned}$$

The first inequality follows by the lemma assumption and the second one by the inductive assumptions for T_u and T_v . The last inequality is a consequence of [Lemma 9](#) and the final equality follows by the additivity of function size .

Recall that we scaled weights so that $\text{ws}(\mathcal{L}(T)) = |\mathcal{L}(T)|$. Therefore, applying (6) to the whole tree T yields $\text{ws}(T) \leq (\text{ws}(\mathcal{L}(T)) + |\mathcal{L}(T)|)^{\log_2(\xi+2)} = (2 \cdot |\mathcal{L}(T)|)^{\log_2(\xi+2)} = (\xi + 2) \cdot |\mathcal{L}(T)|^{\log_2(\xi+2)}$. Hence,

$$\frac{\text{weight}(T)}{\text{weight}(\mathcal{L}(T))} = \frac{\text{ws}(T)}{\text{ws}(\mathcal{L}(T))} \leq \frac{(\xi + 2) \cdot |\mathcal{L}(T)|^{\log_2(\xi+2)}}{|\mathcal{L}(T)|} = (\xi + 2) \cdot |\mathcal{L}(T)|^{\log_2(\xi/2+1)} ,$$

which concludes the proof. ◀

5 Conclusions

We showed a deterministic algorithm ALG for the MPMD problem whose competitive ratio is $O(m^{\log_2 5.5})$. The currently best lower bound (holding even for randomized solutions) is $\Omega(\log n / \log \log n)$ [3]. A natural research direction would be to narrow this gap.

It is not known whether the analysis of our algorithm is tight. However, one can show that its competitive ratio is at least $\Omega(m^{\log_2 1.5}) = \Omega(m^{0.58})$. To this end, assume that all requests arrive at the same time. For such input, OPT does not pay for delays and simply returns the min-cost perfect matching. On the other hand, ALG computes the same matching as a greedy routine (i.e., it greedily connects two nearest, not yet matched requests). Hence, even if we neglect the delay costs of ALG, its competitive ratio would be at least the approximation ratio of the greedy algorithm for min-cost perfect matching. The latter was shown to be $\Theta(m^{\log_2 1.5})$ by Reingold and Tarjan [40].

The reasoning above indicates an inherent difficulty of the problem. In order to beat the $\Omega(m^{\log_2 1.5})$ barrier, an online algorithm has to handle settings when all requests are given simultaneously more effectively. In particular, for such and similar input instances it has to employ a non-local and non-greedy policy of choosing requests to match.

References

- 1 Susanne Albers and Helge Bals. Dynamic TCP acknowledgment: Penalizing long delays. *SIAM Journal on Discrete Mathematics*, 19(4):938–951, 2005.
- 2 Antonios Antoniadis, Neal Barcelo, Michael Nugent, Kirk Pruhs, and Michele Scquizzato. A $o(n)$ -competitive deterministic algorithm for online matching on a line. In *Proc. 12th Workshop on Approximation and Online Algorithms (WAOA)*, pages 11–22, 2014.
- 3 Itai Ashlagi, Yossi Azar, Moses Charikar, Ashish Chiplunkar, Ofir Geri, Haim Kaplan, Rahul Makhijani, Yuyi Wang, and Roger Wattenhofer. Min-cost bipartite perfect matching with delays. 2017. URL: <https://web.stanford.edu/~iashlagi/papers/mbpmd.pdf>.
- 4 Yossi Azar, Ashish Chiplunkar, and Haim Kaplan. Polylogarithmic bounds on the competitiveness of min-cost (bipartite) perfect matching with delays. 2016. URL: <https://arxiv.org/abs/1610.05155>.
- 5 Yossi Azar, Ashish Chiplunkar, and Haim Kaplan. Polylogarithmic bounds on the competitiveness of min-cost perfect matching with delays. In *Proc. 28th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 1051–1061, 2017.
- 6 Yossi Azar, Amir Epstein, Łukasz Jeż, and Adi Vardi. Make-to-order integrated scheduling and distribution. In *Proc. 27th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 140–154, 2016.
- 7 Nikhil Bansal, Niv Buchbinder, Anupam Gupta, and Joseph Naor. A randomized $O(\log^2 k)$ -competitive algorithm for metric bipartite matching. *Algorithmica*, 68(2):390–403, 2014.
- 8 Nikhil Bansal, Niv Buchbinder, Aleksander Mądry, and Joseph Naor. A polylogarithmic-competitive algorithm for the k -server problem. *Journal of the ACM*, 62(5):40:1–40:49, 2015.
- 9 Marcin Bienkowski, Martin Böhm, Jarosław Byrka, Marek Chrobak, Christoph Dürr, Lukáš Folwarczny, Łukasz Jeż, Jiří Sgall, Nguyen Kim Thang, and Pavel Veselý. Online algorithms for multi-level aggregation. In *Proc. 24th European Symp. on Algorithms (ESA)*, pages 12:1–12:17, 2016.
- 10 Marcin Bienkowski, Jarosław Byrka, Marek Chrobak, Łukasz Jeż, Dorian Nogneng, and Jiří Sgall. Better approximation bounds for the joint replenishment problem. In *Proc. 25th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 42–54, 2014.
- 11 Marcin Bienkowski, Jarosław Byrka, Marek Chrobak, Łukasz Jeż, Jiří Sgall, and Grzegorz Stachowiak. Online control message aggregation in chain networks. In *Proc. 13th Algorithms and Data Structures Symposium (WADS)*, pages 133–145, 2013.
- 12 Benjamin Birnbaum and Claire Mathieu. On-line bipartite matching made simple. *SIGACT News*, 39(1):80–87, 2008.
- 13 Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.
- 14 Carlos Brito, Elias Koutsoupias, and Shailesh Vaya. Competitive analysis of organization networks or multicast acknowledgment: How much to wait? *Algorithmica*, 64(4):584–605, 2012.
- 15 Niv Buchbinder, Moran Feldman, Joseph (Seffi) Naor, and Ohad Talmon. $O(\text{depth})$ -competitive algorithm for online multi-level aggregation. In *Proc. 28th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 1235–1244, 2017.
- 16 Niv Buchbinder, Kamal Jain, and Joseph Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *Proc. 15th European Symp. on Algorithms (ESA)*, pages 253–264, 2007.
- 17 Niv Buchbinder, Tracy Kimbrel, Retsef Levi, Konstantin Makarychev, and Maxim Sviridenko. Online make-to-order joint replenishment model: primal dual competitive algorithms. In *Proc. 19th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 952–961, 2008.

- 18 Marek Chrobak. Online aggregation problems. *SIGACT News*, 45(1):91–102, 2014.
- 19 Nikhil R. Devanur and Kamal Jain. Online matching with concave returns. In *Proc. 44th ACM Symp. on Theory of Computing (STOC)*, pages 137–144, 2012.
- 20 Nikhil R. Devanur, Kamal Jain, and Robert D. Kleinberg. Randomized primal-dual analysis of RANKING for online bipartite matching. In *Proc. 24th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 101–107, 2013.
- 21 Daniel R. Dooley, Sally A. Goldman, and Stephen D. Scott. On-line analysis of the TCP acknowledgment delay problem. *Journal of the ACM*, 48(2):243–273, 2001.
- 22 Yuval Emek, Shay Kutten, and Roger Wattenhofer. Online matching: haste makes waste! In *Proc. 48th ACM Symp. on Theory of Computing (STOC)*, pages 333–344, 2016.
- 23 Yuval Emek, Yaacov Shapiro, and Yuyi Wang. Minimum cost perfect matching with delays for two sources. In *Proc. 10th Int. Conf. on Algorithms and Complexity (CIAC)*, 2017. To appear.
- 24 Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497, 2004.
- 25 Bernhard Fuchs, Winfried Hochstättler, and Walter Kern. Online matching on a line. *Theoretical Computer Science*, 332(1–3):251–264, 2005.
- 26 Anupam Gupta and Kevin Lewi. The online metric matching problem for doubling metrics. In *Proc. 39th Int. Colloq. on Automata, Languages and Programming (ICALP)*, pages 424–435, 2012.
- 27 Bala Kalyanasundaram and Kirk Pruhs. Online weighted matching. *Journal of Algorithms*, 14(3):478–488, 1993.
- 28 Anna R. Karlin, Claire Kenyon, and Dana Randall. Dynamic TCP acknowledgement and other stories about $e/(e - 1)$. *Algorithmica*, 36(3):209–224, 2003.
- 29 Anna R. Karlin, Mark S. Manasse, Lyle A. McGeoch, and Susan Owicki. Competitive randomized algorithms for non-uniform problems. *Algorithmica*, 11(6):542–571, 1994.
- 30 Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proc. 22nd ACM Symp. on Theory of Computing (STOC)*, pages 352–358, 1990.
- 31 Sanjeev Khanna, Joseph Naor, and Danny Raz. Control message aggregation in group communication protocols. In *Proc. 29th Int. Colloq. on Automata, Languages and Programming (ICALP)*, pages 135–146, 2002.
- 32 Samir Khuller, Stephen G. Mitchell, and Vijay V. Vazirani. On-line algorithms for weighted bipartite matching and stable marriages. *Theoretical Computer Science*, 127(2):255–267, 1994.
- 33 Elias Koutsoupias and Akash Nanavati. The online matching problem on a line. In *Proc. 1st Workshop on Approximation and Online Algorithms (WAOA)*, pages 179–191, 2003.
- 34 Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing LPs. In *Proc. 43rd ACM Symp. on Theory of Computing (STOC)*, pages 597–606, 2011.
- 35 Aranyak Mehta. Online matching and ad allocation. *Foundations and Trends in Theoretical Computer Science*, 8(4):265–368, 2013.
- 36 Aranyak Mehta, Amin Saberi, Umesh V. Vazirani, and Vijay V. Vazirani. Adwords and generalized online matching. *Journal of the ACM*, 54(5), 2007.
- 37 Adam Meyerson, Akash Nanavati, and Laura J. Poplawski. Randomized online algorithms for minimum metric bipartite matching. In *Proc. 7th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 954–959, 2006.
- 38 Joseph Naor and David Wajc. Near-optimum online ad allocation for targeted advertising. In *Proc. 16th ACM Conf. on Economics and Computation (EC)*, pages 131–148, 2015.

- 39 Yvonne Anne Pignolet, Stefan Schmid, and Roger Wattenhofer. Tight bounds for delay-sensitive aggregation. *Discrete Mathematics & Theoretical Computer Science*, 12(1):39–58, 2010.
- 40 Edward M. Reingold and Robert Endre Tarjan. On a greedy heuristic for complete matching. *SIAM Journal on Computing*, 10(4):676–681, 1981.